Patent Application
Docket No. 92717-00345USP1

MAIL STOP APPLICATION
Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING BY EXPRESS MAIL
"EXPRESS MAIL" Mailing Label
No. **EV227868095US**
Date of Deposit: OCTOBER 21, 2003
I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" on the date indicated above and is addressed to the Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

Type or Print Name: Carol Mitchell

Signature Carol Mitchell

METHOD OF AND SYSTEM FOR RULES-BASED POPULATION OF A KNOWLEDGE BASE USED FOR MEDICAL CLAIMS PROCESSING

CROSS-REFERENCE TO RELATED APPLICATIONS

This patent application is a continuation-in-part of, and incorporates by reference the entire disclosure of, co-pending U.S. Patent Application No. 10/336,104, which was filed on January 3, 2003. U.S. Patent Application No. 10/336,104 is a continuation-in-part of U.S. Patent Application No. 09/859,320, which was filed on May 16, 2001. This patent application incorporates by reference the entire disclosure of U.S. Patent Application No. 09/859,320. This patent application also incorporates by reference the entire disclosure of a U.S. Patent Application entitled "Method of and System for Populating Knowledge Bases using Rules-Based Systems and Object-Oriented Software," filed on the same date as this patent application and bearing attorney docket no. 92717-00344USP1.

BACKGROUND OF THE INVENTION

Technical Field of the Invention

The present invention is generally related to medical claims processing, and more specifically, but not by way of limitation, to a method of and system for rules-based population of a knowledge base used for medical claims processing.

Description of Related Art

The healthcare industry has become very large from the standpoint of financial transactions. Healthcare providers ("providers"), such as hospitals, physicians, and professional service providers (e.g., laboratories, pharmacies), have expanded treatment and services as medicine has become more diverse in treating people for many more ailments than in the past. One reason for the expanded treatment and services includes advancements in research and development of technology to aid physicians in diagnosing and treating patients.

Accordingly, the healthcare insurance industry has grown to assist patients in paying for healthcare expenses. In providing for payment of services, healthcare insurance companies and other payment organizations (e.g., Medicare, Medicaid, etc.) ("payers") have established medical services and procedures to be covered for the treatment of patients. The providers and other organizations (e.g., industry standards groups and government regulators) have developed a variety of protocols to submit payment requests or medical insurance claims ("claims") to the payers for payment or co-payment of treatment and services rendered by the providers.

The protocols that have been developed by the payers and other organizations were developed in an effort to form standards by which payers recognize treatment procedures and services performed. The protocols enable the payers to more easily determine if treatment procedures and services are covered by the insurance policies of patients. As the industry developed, a number of different protocols developed in the way of claim forms, including UB-92 (formerly UB-82), which is utilized by institutional providers (e.g., hospitals), HCFA 1500, which is utilized by professional providers (e.g., physicians), and institutional and professional 837, the newest standard mandated by HIPAA. The claim forms traditionally were in the form of paper. However, the claim forms have evolved with technology and many are now prepared on a computer in an electronic format. While most providers utilize computers to prepare the electronic claim forms, small and rural providers continue to utilize paper claim forms.

Whether the provider utilizes paper or electronic claim forms, the various codes that identify medical diagnosis and treatment that have been generated by healthcare industry standards groups (e.g., National Uniform Billing Committee (NUBC), State Uniform Billing Committee (SUBC), government regulators, the AMA, CMS, payers, etc.), are used in filling out claim forms for submission to payers. By utilizing standardized codes, providers and payers may communicate in a uniform manner. There are approximately 20 different code sets containing nearly 60,000 unique codes that have been developed for providers to utilize based on the specific field of medicine, service, treatment, etc., that is provided to the patient. For example, the International Classification of Disease 9th revision codes, generally known as ICD-9 codes, are utilized to describe diagnoses and the treatment of medical conditions afflicting various body parts (e.g., head, arms, legs, etc.). Other types of codes include Common Procedure Terminology (CPT) codes, which are used for physician codes; Diagnosis Related Group (DRG), which are used for in-patient procedures; and Healthcare Procedure Coding Systems (HCPCS), which are used to report outpatient procedures, drugs, durable medical equipment and outpatient services. As understood in the art, the codes generally are updated annually and new types of codes are created as medical procedures and specialties are formed.

While the code sets have been established to enable the healthcare and insurance industries to use common codes, there are many reasons why reporting problems result in a medical procedure or service not always being easily classified with a particular code or properly reported. A provider may perform a procedure and write or dictate a treatment analysis to be submitted for insurance reimbursement. One claim coder (i.e., individual who interprets the treatment analysis and assigns the proper code into the claim form) may interpret the treatment analysis differently from a different claim coder. And, based on the correctness of, and compliance to, claim submission rules of the claim codes submitted, the payer may or may not approve the procedure or treatment for reimbursement.

FIGURE 1A is a diagram of a sample UB-92 claim form. The claim form includes 85 identified fields for entry of information and/or codes. Various information may be entered into the associated fields. For example, field 1 is used for entry of the provider name, address, and telephone number, as required. Field 3 is used for entry of the patient control number, which is the account number for the patient. As indicated, no special characters (e.g., *, @, -, #, etc.) are allowed. Field 4 indicates the type of bill and is a three-digit code, where the first digit indicates type of provider (e.g., hospital, skilled nursing, home health, etc.), the second digit indicates the type of care (e.g., inpatient, outpatient, specialized services, etc.), and the third digit indicates the type of claim (e.g., non-payment/zero claim, admit through discharge claim, interim-1st claim, etc.). Fields 67-81 are used to enter ICD-9 codes for diagnosis and procedure identification. As indicated, determining the proper codes to insert is often complex and difficult, especially codes relating to the diagnosis and procedure information of the ICD-9 codes. In fact, comprehensive educational courses are provided to medical assistants to teach how the forms are to be properly filled out.

Entry of the UB-92 claim form may be a challenging task due to the complexity of information necessary and to the complexity of the medical codes and insurance information that must be determined and entered. While one may become an expert at entry of the claim form, because each payer has different rules for authorizing payment based on the information submitted on the claim form and because each provider has different methods or procedures for determining the information to be entered into the claim form, the claim submission and reimbursement process often becomes a financial burden for both the provider and payer, delaying payments and increasing healthcare administrative costs.

As well understood in the art, there are large numbers of providers and payers. While there are an estimated 3000(+) major providers and payers, there are several thousands of physicians, all of whom submit claim forms to the thousands of payers. Because patients of a single provider facility may have insurance with many tens or hundreds of payers, the providers are overburdened and practically incapable of maintaining knowledge as to the rules and requirements, addressees, contacts, etc., for each payer. One quickly understands the magnitude of the coordination of communications needed between the providers and payers.

To assist both the provider and payer with the coordination of claim submission, an industry of clearinghouses has developed. FIGURE 1B is a diagram that illustrates an exemplary business model of providers 102a-102d (collectively 102) for submitting claim forms 103 to payers 104a-104d (collectively 104) via clearinghouses 106a-106c (collectively 106). The claim form 103 may be submitted on paper or electronically via data packets 108 across a communication system (See FIGURE 4). As can be seen in FIGURE 1A, the number of communication links between the providers 102 and payers 104 is substantially reduced by the inclusion of the clearinghouses 106.

The clearinghouses 106 perform, at least in part, distribution duties similar to a postal distribution center for the providers of the claim forms 103 in either paper or electronic formats. The clearinghouses 106 perform, at least in part, communication of status (e.g., acceptance, rejection, etc.) of the submitted claims from the payers 104 to the providers 102. The process by which the claims are accepted or rejected by the payers 104 is generally known as the adjudication process.

FIGURE 2 is a diagram that illustrates an exemplary process time line 200 describing general operations for processing a medical claim by the parties of FIGURE 1B. The processing may include preparing, submitting, distributing, adjudicating, and remitting on the claim for the providers 102, clearinghouses 106, and payers 104. As understood in the art, the process starts at step 202 as the claim form 103 is filled out with patient information, such as name, address, phone number, religion, etc., at a pre-admit phase of a patient being processed to see a provider. At step 204, an admission and eligibility phase is performed by the provider determining eligibility for services of a patient and admitting the patient to be treated. The process of admitting the patient may be determined based on, at least in part, the patient having valid insurance and/or credit. The admission/eligibility phase at step 204 may further include the process of provider 102 treating and/or diagnosing the patient.

At step 206, the patient is discharged by the provider 102. The provider 102 may thereupon update a patient chart for the patient with treatment and diagnosis information as understood in the art. The treatment and diagnosis information may be transposed onto the claim form 103 by determining the appropriate codes (e.g., ICD-9 codes) to enter into the correct field(s) (e.g., field 67) of the claim form 103 at step 208. Once the claim form 103 is completed and ready for submission to a payer 104, a "bill drop" or communication of the claim form 103 may be made from the provider 102 electronically or via mail at step 210. In general, the bill drop at step 210 is performed in a batch process to the clearinghouse 106 due to the fact that computer systems of the providers 102 and payers 104 do not have direct communication because the computer systems and software do not share compatible architectures.

Some of the reasons for the computer systems of the providers 102 and payers 104 not having compatible architectures include: (1) the healthcare industry having traditionally performed paper processing transactions, (2) the computer and software systems of the providers 102 and payers 104 never having been developed to communicate with one another, (3) the codes developed for the providers 102 not necessarily having been communicated to and adopted by the payers 104, (4) the clearinghouses 106 having been established to manage and process claims, thereby not having an incentive to adopt a direct, real-time payment system between the providers and payers, (5) the payers having limited incentive to expedite payment as delay in payment increases interest revenue for the payers, (6) the number of people, organizations and government entities defining codes adding layers of complexity to the process, and (7) technology not having been fully adopted by the healthcare industry. For example, there are very few direct connections between trading partners (i.e., specific provider 102 and payer 104).

Software developers and information technology companies that the providers 102 and payers 104 have utilized to develop systems and software to manage the claims processing have generally been devoted to either the provider 102 or payer 104, so that the concerns of the other side essentially have been unincorporated in the development process. In other words, the business model for the systems have focused on either the payer 104 or provider 102 side in terms of collecting revenue. On the provider side, the systems are established to conform to the needs of the general population of payers 104 (e.g., to form submission compliance with as many payers 104 as possible), which typically causes the systems to be less compatible with any specific payer 104. On the payer side, the systems are established to conform to the needs of the specific payer systems without regard to the capabilities of the providers' systems (e.g., to receive form submission from as many providers 102 as possible), which typically causes the systems to be less compatible with any specific provider 102.

While the incompatibility of the systems of the providers 102 and payers 104, and lack of desire and motivation of the payers 104 have held back progress in improving the technology for the healthcare industry to more efficiently and effectively process claims, the major problems that the industry has to overcome include, but are not limited to, the (i) dynamic environment of rapidly changing codes, (ii) conflicting reporting requirements, and (iii) contradictory claim filing guidance. These problems and turmoil have resulted in a complete industry being created to focus on interpreting the changes in codes and reporting guidance and creating software programs to evaluate the contents of the claim forms 103 and assess the validity of the claim forms 103 before being sent to the payer 104 for adjudication and settlement. This industry, which includes clearinghouses 106, receives change notifications and error reports in many different forms. In many cases, an originator of the change announces how the change should be handled by payers and fiscal intermediaries. These change handling instructions are referred to as "edits" as understood in the healthcare industry or various other industries.

Continuing with FIGURE 2, the process of applying edits to submitted claim forms is performed at step 212. This process is performed by the clearinghouse 106 for each of the claims submitted in a batch, which may include large numbers (up to 500 or more) of claims. Edits may come in many forms, including being (i) tucked into the body of a government released transmittal, (ii) listed in a spreadsheet or table containing hundreds or thousands of edits that have been created by both providers 102 and payers 104, and (iii) contained in the text of specification documents such as, for example, the X12 Institutional 837. In many cases, edits are created by provider organizations in order to overcome a shortfall in a legacy accounting system that cannot be modified to accommodate new changes. Regardless of the source for the edits, the edits are almost always provided in free form English language text. Because the edit text is generated by different individuals, in different locations, at different times, often using different sentence structures, and because of the nature of the edit generation process, the. task of analyzing, cataloging, and managing edits has become a time and labor-intensive activity. One example of the complexity of managing edits is a healthcare management company having 100 provider institutions located in ten different states submitting medical insurance claims to Medicare, ten different Medicaid payers, an undetermined number of commercial payers, and Civilian Health and Management Program Uniformed Service (CHAMPUS), which recently became Tricare, for providing medical insurance to military dependents and retired military personnel, thereby resulting in the healthcare management company having 10,000 or more edits to manage.

The term “edits” historically was used to describe the process of correcting information in a data file. While the edits still refer to correcting information, the term "edits" in the healthcare industry for insurance claims provides for a directive to correct information that is incorrect or to reject claims that do not comply with business rules established by a payer. In other words, the edits may be considered statements of situations that cause an error to occur to hinder payment or processing of final adjudication of a particular insurance claim. The business rules of payer 104, which may be established arbitrarily or based on the policies of the payer 104, for example, may be established and modified on an annual basis or more frequently. For example, one business rule may be as simple as requiring the last name to be entered with all capital letters. Another business rule may indicate that a certain procedure is to be denied reimbursement if a certain diagnosis not requiring the procedure to be performed is reached. Yet another business rule may require a certain identifier in a field if an intern assists in a medical procedure. And, if any of these business rules are violated, an edit is generated and applied to the insurance claim form 103 to notify the provider 102 that a correction is needed per the instructions of the edit.

An example of an edit includes the following: "Move UPIN from 82AA to 82BA," where UPIN is an abbreviation for universal provider identification number and AA and BA are field identifiers in the form locator referring to an attending physician on a UB-92 claim form 103. As understood in the art, the word "move" alternatively may be written as "copy," "change," include," or other synonym in an edit. The choice of words for an edit is arbitrary as there are no particular standard terms to be used for edits for the individuals who generate the edits. And, because of the multitude of different, yet related terms, the edits associated with the business rules may mean the same thing or be substantially semantically similar and unnecessarily increase the overall number of edits to which the providers must adhere.

The end goal for the providers 102 is to expedite final adjudication of medical claims by minimizing rejection of the medical claims by the payer 104 due to errors entered on the claim forms 103. To minimize the errors entered on the claims form 103, an understanding of the edits is desirable as the edits offer a roadmap for mistakes that may be made in view of the business rules and codes utilized to adequately and correctly complete the claim forms 103 for claim validation. Claim validation occurs when at least the following items are satisfied: (i) the claim form 103 is complete in the eyes of the payer 104, (ii) the data are properly formatted in the proper location on the claim form 103, and (iii) the data accurately reflects medical services provided and meets service constraints, which are generally embodied in the edits. However, because of the large volume of edits and frequency of edit modifications, creation of a complete understanding and knowledge base of the tens of thousands of edits is substantially impossible for an individual attempting to design a system to expedite medical claims processing.

After the process of applying edits to a claim form at step 212, if there are no edits applied to the claim form 103 because no errors were detected, then the claim form 103 is communicated to the payer 104 in a format that the payer 104 requires. Otherwise, if errors were detected on the submitted claim form 103, then the claim form 103 and associated edits 215 are communicated back to the provider 102 for correction to the claim form 103.

At step 216, the payer 104 receives the claim form 103. A receipt of receiving the claim form 103 may be communicated back to the provider 102 via the clearinghouse 106 for notification purposes. At step 218, the payer 104 adjudicates on approving the claim for payment purposes. The adjudication is based on rules or policies that the payer 104 may have for the health insurance plan of the patient. Generally, the edits include enough of the policies so that the claims are approved by the payer 104, but is not always the case.

At step 220, a status including the results of the adjudication process of step 218 may be communicated via the clearinghouse 106 back to the provider 102. If the claim was rejected, the provider 102 may be allowed to cure the defect. Additionally and/or alternatively, the provider 102 or patient may appeal the rejection at this stage without having to resubmit another or amended claim form 103. If the claim was approved, then payment 223 of the claim may be resubmitted to the provider 102, either directly or indirectly. At step 224, the provider 102 receives the payment 223 and applies it to collections at step 226. At this point, the claim is considered closed as payment by the payer 104 has been tendered.

SUMMARY OF THE INVENTION

A method of populating a knowledge base includes filtering a translated edit via at least one rule to determine a match between a syntax of the translated edit and a syntax of the rule, executing a method call responsive to the filtering step resulting in at least one match being determined, and populating the knowledge base responsive to the executing step.

An article of manufacture for populating a knowledge base used in validating medical claims includes at least one computer readable medium and processor instructions contained on the at least one computer readable medium. The processor instructions are configured to be readable from the at least one computer readable medium by at least one processor and thereby cause the at least one processor to operate as to filter a translated edit via at least one rule to determine a match between the translated edit and the rule, execute a method call responsive to the translated edit having been filtered and a determination of at least one match, and populate the knowledge base responsive to the execution of the method call.

A system for populating a knowledge base includes means for filtering a translated edit via at least one rule to determine a match between a syntax of the translated edit and a syntax of the rule, means for executing a method call responsive to the filtering having resulting in at least one match having been determined, and means for populating the knowledge base responsive to an output of the means for executing.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the principles of the present invention may be obtained by reference to the following Detailed Description when taken in conjunction with the accompanying Drawings wherein:

FIGURE 1A, previously described, is an exemplary claim form used in a medical claim process;

FIGURE 1B, previously described in part, is a diagram that shows an exemplary business model of providers submitting claim forms to payers via clearinghouses as understood in the art;

FIGURE 2, previously described in part, is a diagram illustrating an exemplary process line describing general operations for processing a medical claim by the parties of FIGURE 1B;

FIGURE 3 illustrates an exemplary set of objects as understood in the art that may be utilized in consolidating edits for the medical claims submission process as shown in FIGURE 2;

FIGURE 4 is an exemplary block diagram of a distributed network operable to enable the entities of the medical industry of FIGURE 1B to electronically perform medical insurance claim submission, processing, and adjudication;

FIGURE 5 is an exemplary block diagram model describing a ruled-based system for generating the consolidated edit list for processing claims by the healthcare entities of FIGURE 1B;

FIGURE 6 is an exemplary flow diagram that generally describes a process for consolidating edits to form a consolidated edit list to be utilized in improving the efficiency of the claims filing process for the healthcare entities of FIGURE 1B;

FIGURE 7 is an exemplary process diagram describing the process of forming the consolidated edit list to more efficiently process claims submitted on the distributed network of FIGURE 4;

FIGURE 8 is a flow diagram that illustrates an overall flow for populating a knowledge base and validating claims using knowledge in the populated knowledge base; and

FIGURE 9 is a flow diagram that illustrates in more detail step 804 of FIG. 8.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS OF THE INVENTION

Medical insurance claims processing has become a significantly complex, expensive, and time consuming part of the medical industry due to the vast number of treatment codes, rules, and edits (i.e., claim form change instructions written in English language form) that have been formed over the years by governmental entities, medical industry standards committees, medical providers 102, and payers 104. The frequency of changes of the treatment codes continues to cause additional edits to be generated to resolve claim form entries that do not conform to the rules or errors that the payers 104 reject as being non-conforming or impermissible due to the rules of the payer 104 for the medical procedure or treatment to be reimbursed. In submitting medical insurance claims, the providers 102 generally send the claim forms 103 in batch electronically or via mail to payers 104 via a clearinghouse 106 (See FIGURE 2). The clearinghouse 106 processes the claim forms 103, either manually or automatically, and distributes the claim forms 103 to the designated payers 104. If the claim form 103 is properly filled out, then the payer 104 adjudicates whether or not to pay for the medical procedure. If the claim form 103 is not properly filled out, then one of thousands of edits may be applied to the claim form 103 by the clearinghouse 106 or payer 104 and returned to the provider 102 for correction. Each payer 104 may have different rules and each time the treatment codes are updated, new rules and edits are created, which makes the process of filling out the claim form 103 increasingly more difficult, as discussed in, for example, Application No. 10/336,104.

In improving the efficiency of applying the edits to the claim forms 103, the total number of edits are consolidated by performing a comparison between the words of each of the edits to determine the degree of similarity of the words and/or semantics of the edits being compared. In one embodiment, a token list composed of each word of an edit string is generated for each edit being compared. A comparison may be made between the token lists and a percentage or discrete value may be formed based on the number of matched words divided by the average number of words in the edits being compared to indicate the degree of semantic similarity between the two edits. A scale may utilize the discrete value and a symbolic description may be generated based on the discrete value to indicate the degree of similarity between the two edits. In one embodiment, the scale is a Likert scale, whereby ranges of values (e.g., between 0.0 and 1.00 percent) are utilized to categorize the degree of similarity with the symbolic description. The symbolic description may include text, such as "significant similarity," "high similarity," "moderate similarity," "marginal similarity," and "insignificant similarity." Alternatively, grades (e.g., "A," "B," "C," etc.) or values (e.g., 100, 95, 90, etc.) may be generated as the symbolic description.

A sorted list of edits may be generated based on the discrete value and/or symbolic description. In one embodiment, the sorted list may be sorted in descending order from most similar semantically to least similar. The edits that are most similar may be consolidated by using a rule that uses the symbolic description in consolidating or recommending consolidation of the edits. In consolidating the edits, a union of the token lists of the edits being consolidated may be formed to a single edit that describes the edits determined to have a "significant similarity," for example. In one embodiment, a predetermined threshold may be utilized. For example, edits having a similarity within ten percent may be considered significantly similar and be automatically consolidated. Alternatively and/or additionally, edits considered to have a "high similarity" and /or "moderate similarity" may be recommended to be manually inspected for the edit consolidation process. Edits that are determined to have "marginal similarity" and/or "insignificant similarity" may be treated as edits that are not combinable with other edits. For example, edits that are determined to be above a predetermined threshold of similarity, such as 60 percent, may be formed as separate edits. By consolidating the edits, a manageable number of edits may be produced to be applied to rejected claim forms 103 being submitted by the providers 102 for reimbursement by the payers 104. By applying a reduced number of edits to the rejected claim form 103, the efficiency of processing of the claim forms may be improved to minimize or eliminate the need to submit claim forms 103 as a batch process. Depending upon how the similarity is measured, the number of edits may be reduced from thousands to hundreds.

FIGURE 3 is a diagram that illustrates an exemplary set of objects 300 as understood in the art that may be utilized in consolidating edits for the medical claims submission process 200 as shown in FIGURE 2. An edit object 302 may include a number of data elements 304 to describe an edit and information associated with other elements. For example, "editString" 306 stores the English text description of the edit and the "similarEdits" 308 stores the edits or links to edits that are determined to be similar. Additionally, the edit object 302 may be associated with functional objects 310 used in the edit comparison process according to the principles of the present invention.

Other objects may be utilized to describe or model the entities associated with medical claims processing. For example, objects "InstitutionalProvider" 312 and "Payer" 314 may be utilized to describe particularities of different providers 102 and payers 104. In one embodiment, an instance of the payer object may include rules that a payer 104 has for adjudicating submitted claims and edits associated therewith based on the rules of the payer 104. Accordingly, by having objects that model the different entities (e.g., provider 102 and payer 104) with the claims filing process 200, more specific information for consolidating the edits may be generated. Additionally and/or alternatively, the entity objects 312 and 314 may be applied to a distributed network (see, for example, FIGURE 4) and be utilized to process claim forms 103 being filed in real-time or otherwise.

FIGURE 4 is an exemplary block diagram of a distributed network 400 operable to enable the entities of the medical industry of FIGURE 1B to electronically perform medical insurance claim submission, processing, and adjudication. A provider server 402 may be utilized for entering information into a claim form 103 and electronically submitting the claim form 103 to a payer server 404 via a clearinghouse server 406 over a network 408 as understood in the art.

In another embodiment, the provider server 402 may submit the claim form 103 directly to the payer server 404 via the network 408. The network 408 may be the Internet, satellite network, wide area network (WAN), telecommunications network, or other communication system.

The provider server 402 may include a processor 410 coupled to a memory 412, input/output (I/O) unit 414, and storage unit 416. The processor 410 may operate software 418 for entry of patient information into the claim form 103. The software 418 may be object oriented such that the information is applied to elements of an object to be communicated over the network 408 for processing via the I/O unit 414. The memory 412 may store the software 418 prior and/or during execution. The storage unit 416 may store a database 420 that maintains the claim forms 103 in an object oriented or other format. The I/O unit 414 of the provider server 402 may communicate the claim form 103 using data packets 422 or other communication protocol as understood in the art.

The clearinghouse server 406 may operate to receive the claim form 103 via the network 408. In one embodiment, the clearinghouse server 406 receives claim forms 103 in a batch process. However, the principles of the present invention may provide for receiving claim forms 103 on an individual basis. The clearinghouse server 406 generally includes the same hardware as the provider server 402, including a processor 424 coupled to a memory 426, I/O unit 428, and storage unit 430. The processor 424 may execute software 432 that receives the information of the claim form 103. If the claim form 103 is represented as an object, then the software 432 may process the claim utilizing objects to verify the information of the claim form 103. In processing the claim, claim verification rules in the form of objects that specify how content is to be submitted on the claim form 103 in order to be adjudicated by one or more payers 104 may be utilized.

In one embodiment, the clearinghouse server 406 may apply the claim verification rules to the submitted claim form 103 and, in the case of an error (i.e., information entered into the claim form 103 not complying with the claim verification rules), apply or associate one or more edits to the claim form 103. The edit(s) applied to the claim form 103 may be based on edits specific to the payer 104 that the claim form 103 is being submitted or based on a reduced set of edits according to the principles of the present invention. The results of the processing of the claim form 103 may be stored in a database 434 with or without the claim form 103 in an object oriented or other format. The I/O unit 428 may communicate the processed claim form 103 and any associated edits or other information to the payer server 404 via the network 408 in data packets 436.

The payer server 404 may include generally the same hardware as the provider server 402, including a processor 438 coupled to a memory 440, I/O unit 442, and storage unit 444. The processor 438 may execute software 446 to process the claim forms 103 received for adjudication by the payer 104. The I/O unit 442 may receive the claim form 103 via data packets 436 and communicate the claim form 103 to the processor 438. The software 446 may utilize objects to verify the information of the claim form 103 and associated edit(s), if any, to adjudicate the validity of the claim for reimbursement to the provider 102 or patient. In accordance with the principles of the present invention, a consolidated set of edits may be utilized to reduce processing time. Alternatively, a reduced set of edits specific to the payer 104 may be utilized. A database 448 may be utilized to store (i) the submitted claim form 103, (ii) results of the verification, and (iii) results of the adjudication process in object oriented or other format. Accordingly, a rejected claim may be communicated to the provider server 402 via the clearinghouse server 406 or directly to the provider server 402 by the I/O unit 442. It should be understood that other or additional servers may be utilized in accordance with the principles of the present invention to generate and/or apply the consolidated set of edits to the submitted claim forms 103.

FIGURE 5 is an exemplary block diagram model describing a rule-based system 500. A user 502 may operate the rule-based system 500, which may include an expert system shell 504 having high-level modules that are generally included in rule-based systems as understood in the art. The expert system shell 504 may include a user interface 506, explanation system 508, inference engine 510, and knowledge base editor 512.

Almost all expert systems also have an explanation system 508, which allows the system 500 to explain its reasoning to the user 502. The inference engine 510 may be utilized to generate knowledge of the information provided thereto as understood in the art. The explanation system 508 and the inference engine 510 may be coupled to case specific data 514 containing specific information related to the problem being solved (e.g., reduction in the number of edits). The knowledge base editor 512 helps the expert or knowledge engineer to easily update and check a knowledge base 516 that is provided or formed. The knowledge base 516 may be made up of a collection of facts and rules that constitute the information model usable by the system 500.

In operation, the user 502 interacts with the system 500 via a user interface 506 that may be graphical (e.g., graphical user interface (GUI)), natural language or any other style of interaction. The inference engine 510 may be used to reason with both the expert knowledge (extracted from an expert in the field) and data specific to the particular problem being solved. The inference engine 510 may allow the user 502 to test rule conditions and add new information for the system 500 to use as it is executing. The expert knowledge is typically in the form of a set of IF-THEN rules. The case specific data 514 includes both data provided by the user 502 and partial conclusions (along with certainty measures) based on this data. In a simple forward chaining rule-based system, the case specific data are the elements in working memory.

One feature of expert systems is the way domain specific knowledge is separated from more general purpose reasoning and representation techniques. The general purpose reasoning may be referred to as the expert system shell 504. Given a new kind of problem to solve (e.g., reduction in edits), an expert system shell 504 that provides the right sort of support for that problem may be composed, so that expert knowledge may be provided to solve the problem. As understood in the art, commercially available expert systems are available to reduce development time.

The system 500 may be executed on one of the servers of the distributed network 400. Alternatively, the system 500 may be executed on an independent computing system (not shown). The processes of FIGURES 6 and 7 may be utilized with the rule-based system 500.

FIGURE 6 is an exemplary flow diagram 600 that generally describes a process for consolidating edits to form a consolidated edit list to be utilized in improving the efficiency of the claims filing process 200 for the healthcare entities of FIGURE 1B. The consolidation process starts at step 602. At step 604, an edit list is accessed. The edit list may be a complete edit list encompassing all edits possible in the medical field, a partial edit list encompassing edits possible in a medical specialty, or a partial edit list encompassing edits applicable to one or more payers 104. Other complete or partial lists may be utilized for consolidation according to the principles of the present invention. In accessing the edit list, the edit list may be stored in a database (e.g., database 430 or 424) and read into memory, received via a communication, or gathered by querying one or more storage units or computing devices maintaining lists of edits.

At step 606, words of at least two edits are compared to determine similarity and/or semantic similarity. The comparison may be performed as understood in the art. One embodiment may include forming tokens for each word of an edit and comparing the tokens. The comparison may include comparing a single edit to each other edit of the edit list. The similarity may be a degree of similarity based on the intersection of the words being compared. Alternatively, the similarity may be based on the words and synonyms of those words. Still yet, words being substantially semantically similar (e.g., copy and move) may be used for the comparison. The degree of similarity may be a discrete value indicative of a ratio or percentage formed by the intersection of words as determined by the comparison divided by the total number of words of the two edits being compared. A scale may be utilized in the comparison process to establish the degree of similarity and, optionally, a symbolic description may be applied to the edits for later usage in consolidating similar edits. In one embodiment, a Likert scale as shown in TABLE 1 may be utilized in forming the symbolic description. The edits may be sorted by similarity to assist in the consolidation process.

Range (percent)	**Symbolic Description**
91-100	Significant Similarity
81-90	High Similarity
61-80	Moderate Similarity
40-60	Marginal Similarity
0-20	Insignificant Similarity

TABLE 1. Likert Scale

At step 608, a new or single edit may be formed based on the similarity between the at least two edits. The single edit may include at least a portion of each edit being consolidated into the single edit. In other words, the single consolidated edit may be the union of two or more edits being combined to form the single edit. The consolidated edit may be written to the database 434. The process ends at step 610.

FIGURE 7 is an exemplary object interaction diagram 700 describing the process of forming the consolidated edit list to more efficiently process claims submitted on the distributed network 400 of FIGURE 4. A user 502 may utilize the process composed of software to form the consolidated edit list. In one embodiment, the software is coded in an object oriented format to represent or model the edits, rules, fields, etc., in an object format in FIGURE 3.

Two instances of the edit object 302 may be formed as Edit1 object instance 704 and Edit2 object instance 706 (i.e., the edit objects 704 and 706 are instantiated). In one embodiment, to form the edit objects, edits are inserted into a hash table keyed to the edit strings and the two edit object instances 704 and 706 may be generated with edit strings from the hash table keys. The edit consolidation process 600 starts at step 608, where the user commands the

Edit1 object instance 704 to build a list of words from the edit string associated with the object. In one embodiment, the list of words is tokens containing each word of the edit string. At step 710, the user commands the Edit2 object instance 706 to build a list of words from the edit string associated with the object. The command may instruct some or all of the edit objects that are to be compared to the Edit1 object instance 704.

EXHIBIT 1 provides an exemplary instance of an edit object 302 of FIGURE 3 formed from an exemplary Medicare edit. The instantiated Medicare edit (i.e., instance of the Medicare edit object) of EXHIBIT 1 is designated MEDICARE-EDIT 576968652 and includes the edit string "FOR MEDICARE INPATIENT CLAIMS TOB 12X AND REVENUE CODE 42X EXISTS THEN OCCURRENCE CODE 29 MUST EXIST," which, in essence, specifies the correction to be made if a claim validation rule is violated by the information entered into the claim form 103.

```
#<db-instance MEDICARE-EDIT 576968652> is a MEDICARE-EDIT
ERROR-CODE                NIL
EDIT-CODE                 NIL
EFFECTIVE-DATE            NIL
SUPERSESSION-DATE         NIL
BILL-ELEMENT              NIL
COMMENT                   NIL
BILL-TYPE                 NIL
VALID-DATA                NIL
CARDINALITY               0
UB-92-COLUMN              "NA"
EDIT-RULES                NIL
EDIT-FACTS                NIL
SUPERSEDING-EDIT          NIL
EDIT-CATEGORY             NIL
EDIT-STRING                "FOR MEDICARE INPATIENT CLAIMS TOB 12X AND REVENUE
CODE 42X EXISTS THEN OCCURRENCE CODE 29 MUST EXIST"
PAYER                     NIL
METHOD-OF-SUBMISSION      NIL
PATIENT-TYPE              NIL
837-DATA-SEGMENT          NIL
ORIGINATOR                "SLU" ........
```

EXHIBIT 1. Instantiated Medicare Edit

At step 712, the user 502 commands the instantiated Edit1 object 704 to compare the edit to other edits (i.e., compare the edit strings via the tokens or token list to tokens of other edits). The Edit1 object 704 compares the associated tokens with tokens of the Edit2 object 706 at step 714. In making the comparison, the software (e.g., software 432) may utilize an inference engine or an instance of a pattern matcher class to assist in rule processing, where rules and assertions may be maintained in assertion and rule attributes of the edit objects 704 and 706. During edit comparison, each edit or edit object 704 and 706 assesses the similarity of the associated tokens to the tokens of the other edit being compared. In one embodiment, the Likert scale is used in converting the ratio or percentage of similarity to a symbolic representation of the similarity (See, for example, TABLE 1).

Each edit involved in a comparison with other edits creates facts that indicate the symbolic similarity to other edits. EXHIBIT 2 provides examples of results of comparisons of edits. As shown, each of the edits have ratios above 0.81 (81 percent) and below 0.90 (90 percent), thereby indicating that the edits all have "high similarity" to the edit that is performing the comparison.

```
(#<db-instance MEDICARE-EDIT 576977460> 0.8571428571428571)
(#<db-instance MEDICARE-EDIT 576981212> 0.8235294117647058)
(#<db-instance MEDICARE-EDIT 576972788> 0.8125)
(#<db-instance MEDICARE-EDIT 576961092> 0.8235294117647058)
(#<db-instance MEDICARE-EDIT 576996484> 0.8823529411764706)
```

EXHIBIT 2. Edit Comparison Ratios

At step 716, the Edit1 object 704 initiates the process of creating edit facts. EXHIBIT 3 provides examples of similarity facts for the edits that are created based on results of comparisons of different edits. The facts may be in the form of "<Edit 1> has ____(degree of similarity)____ similarity to <Edit 2>." Where (Degree of similarity) is an arbitrarily assigned value similar to those found in a Likert Scale. For example a similarity between .9 and 1.0 may be considered HIGH, a similarity between .8 and .89 may be considered MODERATE, a similarity between .6 and .79 may be considered MARGINAL, and a similarity between .0 and .59 may be considered MINIMAL. As indicated, the edit object performing the comparison is MEDICARE-EDIT 576968652. The MEDICARE-EDIT objects being compared are 576961092, 576996484, 576972788, 576981212, and 576977460. "SLU" is the code used to represent the provider, or originator of the edit 104.

```
EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576961092> FOR "SLU"

EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576996484> FOR "SLU"

EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576972788> FOR "SLU"

EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576981212> FOR "SLU"

EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576977460> FOR "SLU"
```

EXHIBIT 3. Examples of Similarity Facts Based on Comparison

The facts may be stored in assertion attributes of an inference engine object. In addition, the inference engine adds assertions provided in EXHIBIT 4. The assertions provide actions for handling different degrees of similarity. For example, edits with "high similarity" are to be automatically consolidated and edits with "marginal similarity" are to be manually reviewed.

```
Consolidate edits with high similarity.
Consolidate edits with significant similarity.
Manually-review edits with marginal similarity.
Separately-implement edits with slight similarity.
Separately-implement edits with no similarity.
```

EXHIBIT 4. Assertions for Handling Edits

At step 718, the Edit1 object 704 initiates populating similar edits into the Edit1 object 704. In populating the similar edits into the Edit1 object 704, the inference engine loads a single rule of the form shown in EXHIBIT 5. The rule determines how edits are to be processed. The edit class has methods defined for each of the actions.

```
If <edit-1> has <level> similarity to <edit-2> and <action> edits with
<level> similarity then <action> <edit-1> <edit-2>
```

EXHIBIT 5. Rule

By utilizing the rule, the instantiated edit object of EXHIBIT 1 has a "EDIT-FACTS" attribute updated to include the edit strings of the similar edits as shown in EXHIBIT 6.

```
#<db-instance MEDICARE-EDIT 576968652> is a MEDICARE-EDIT
ERROR-CODE                NIL
EDIT-CODE                 NIL
EFFECTIVE-DATE            NIL
SUPERSESSION-DATE         NIL
BILL-ELEMENT              NIL
COMMENT                   NIL
BILL-TYPE                 NIL
VALID-DATA                NIL
CARDINALITY               0
UB-92-COLUMN              "NA"
EDIT-RULES                NIL
EDIT-FACTS
((EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY
TO EDIT #<db-instance MEDICARE-EDIT 576961092> FOR "SLU")
(EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576996484> FOR "SLU")
(EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576972788> FOR "SLU")
(EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576981212> FOR "SLU")
(EDIT #<db-instance MEDICARE-EDIT 576968652> FOR "SLU" HAS HIGH SIMILARITY TO
EDIT #<db-instance MEDICARE-EDIT 576977460> FOR "SLU"))
SUPERSEDING-EDIT          NIL
EDIT-CATEGORY             NIL
EDIT-STRING               "FOR MEDICARE INPATIENT CLAIMS TOB 12X AND
REVENUE CODE 42X EXISTS THEN OCCURRENCE CODE 29 MUST EXIST"
PAYER                     NIL
METHOD-OF-SUBMISSION      NIL
PATIENT-TYPE              NIL
837-DATA-SEGMENT          NIL
ORIGINATOR                "SLU"
```

EXHIBIT 6. Updated Edit Object with EDIT-FACTS

At step 720, a consolidate method may sort edits in descending order of similarity and consolidate the token lists of the initial edit and the most similar edit at step 722. It may then iterate over the remaining similar edits identifying the different tokens that are to be accounted for in the single new edit being created. As shown in EXHIBIT 7, exemplary rules may be generated based on the rule (EXHIBIT 5) and assertions (EXHIBIT 4)

```
Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576937628>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576925580>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576951716>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576932428>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576962876>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576934508>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576936356>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576951116>).

Rule   EDIT-ACTION-RULE   indicates   (CONSOLIDATE   #<db-instance   MEDICARE-EDIT
576925060> #<db-instance MEDICARE-EDIT 576972268>).
```

EXHIBIT 7. Rules Generated by Rule and Assertions

Based on the rules and consolidation assertions, a union forming a single edit using the token lists (i.e., words of the edit strings) produces a single edit as shown in EXHIBIT 8. The single edit in this case is constructed from five edits, thereby reducing the processing of the edits by a factor of 80 percent. A recommendation of the single, consolidated edit may be determined by the Edit1 object instance 704 at step 724 and recommended to the user 502 at step 726.

```
(35 IS 50 VALUE |42X| 11 |44X| |22X| FOR MEDICARE INPATIENT CLAIMS TOB |12X|
AND REVENUE CODE |43X| EXISTS THEN OCCURRENCE CODE 17 MUST EXIST)
```

EXHIBIT 8. Single Edit Formed from Similar Edits

The single edit may be utilized in the claims process by any of the entities, including provider 102, clearinghouse 106, or payer 104 to increase efficiency in processing the claims. By increasing the efficiency of processing the claims, the provider 102 may receive payment on the claims in a shorter period of time, thereby increasing the revenue stream to the provider 102.

FIGURE 8 is a flow diagram that illustrates an overall flow for populating a knowledge base and validating claims using knowledge in the populated knowledge base. The knowledge base may be, for example, the knowledge base 516 of the system 500. A flow 800 begins at step 802, at which step both consolidated and non-consolidated edits are translated. In order to be able to more easily work with the edits, a translation grammar is used in the edit-translation 802. The translation grammar is used to translate the edits into translated edits that may more readily be filtered by a set of knowledge-base-population rules.

The translation grammar includes a set of patterns that may be used to express relationships that are described in the edits and also specifies circumstances under which a particular claim element needs to have a certain value (i.e., code element) when some other value (i.e., code element) is reported on a claim. For example, a particular edit might state that if a particular revenue code is reported on a claim of a particular claim type and a particular occurrence code is not reported on the claim, the claim should be rejected. The translation grammar permits the relationship between the revenue code and the occurrence code stated in the edit to be explicitly and positively stated. The relationship for the above-described example would be that the revenue code requires the occurrence code on the particular claim type. The translation grammar would therefore result in a translated edit that reflects this requirement. The translation grammar includes identification of particular claim-element locations that must be populated under certain circumstances. The translation grammar also allows for abbreviations to be applied to the name of a code.

More specifically, the translation grammar facilitates the understanding of medical claims processing by creating a specific vocabulary and grammar suitable for use by computers and rule-based systems. The vocabulary is composed of a combination of words that are unique to the healthcare industry (e.g., condition code, occurrence code, form locator, patient, provider, etc.). In addition to the healthcare unique vocabulary, standard English language words are also used to communicate in business and informal daily interactions. The English language words are generally adjectives, nouns, verbs, and prepositions. Some examples of English language words used in the translation grammar are as follows:

Nouns: revenue code, service line, claim element, type of bill;

Verbs: requires, allows, excludes;

Prepositions: on, for, in, from, to;

Adjectives: value, range.

The nouns of the translation grammar are used to describe items of interest in the healthcare claim processing domain. The verbs describe relationships between the nouns in the domain. The translation grammar prepositions constrain the relationships described by the verbs and the adjectives are used to enhance the descriptions of the nouns.

The translation grammar describes legal patterns of usage for the vocabulary set forth above. The translation grammar is an adaptation of American English grammar patterns for declarative sentences. Each translation sentence is composed of a noun phrase, verb phrase and optional prepositional phrases. For example, the translation sentence "Revenue Code value 0420 requires Occurrence Code value 11 on TOB 14X" includes the noun phrase "Revenue Code value 0420," the verb phrase "requires Occurrence Code value 11," and the prepositional phrase "on TOB 14X."

In addition, the translation grammar provides a mechanism that allows a computer to identify inconsistencies and assist a human operator in reducing such inconsistencies. For instance, the exemplary untranslated sentence "Reject Medicare outpatient if revenue code 42X exists and not occurrence code 11, 29, 35" is translated to three separate translations as follows:

Revenue Code value 42X requires Occurrence Code value 11 on TOB outpatient for payer Medicare.

Revenue Code value 42X requires Occurrence Code value 29 on TOB outpatient for payer Medicare.

Revenue Code value 42X requires Occurrence Code value 35 on TOB outpatient for payer Medicare.

A second exemplary untranslated sentence "Reject revenue code 42X exists and not value code 50" is translated into "Revenue Code value 42X requires Value Code value 50." The first exemplary untranslated sentence yields three translated sentences with each sentence including a noun phrase, a verb phrase, and two prepositional phrases. However, the second exemplary untranslated sentence yields one translated sentence with one noun phrase and one verb phrase. As shown by these examples, the first exemplary translated sentences are more precise and more restrictive than the second translated sentence. The translation grammar aids a human operator in identifying and correcting these inconsistencies as noted above.

A manual translation using the translation grammar is typically performed, although an automated translation may also be performed. The translation is performed on both the consolidated edits as well as individual edits that have required manual analysis.

Following the edit translation of step 802, a file is typically produced that contains translated edits and their corresponding untranslated edits. Below is an exemplary excerpt of a translated edit file:

```
(
 (medicare-edit
  (
   (payer "CA BC")
   (token-lists ((VALUE CD 1 REQUIRED TO BILL PVT ROOM)))
   (translation ((RC VALUE 110 REQUIRES VC VALUE 1 TOB IP) (RC
VALUE 111 REQUIRES VC VALUE 1 TOB IP) (RC VALUE 112 REQUIRES VC
VALUE 1 TOB IP) (RC VALUE 113 REQUIRES VC VALUE 1 TOB IP) (RC
VALUE 114 REQUIRES VC VALUE 1 TOB IP) (RC VALUE 115 REQUIRES VC
VALUE 1 TOB IP) (RC VALUE 116 REQUIRES VC VALUE 1 TOB IP) (RC
VALUE 117 REQUIRES VC VALUE 1 TOB IP) (RC VALUE 118 REQUIRES VC
VALUE 1 TOB IP) (RC VALUE 119 REQUIRES VC VALUE 1 TOB IP) (RC
VALUE 119 REQUIRES VC VALUE 1 ON TOB IP) (RC VALUE 118 REQUIRES
VC VALUE 1 ON TOB IP) (RC VALUE 117 REQUIRES VC VALUE 1 ON TOB
IP) (RC VALUE 116 REQUIRES VC VALUE 1 ON TOB IP) (RC VALUE 115
REQUIRES VC VALUE 1 ON TOB IP) (RC VALUE 114 REQUIRES VC VALUE 1
ON TOB IP) (RC VALUE 113 REQUIRES VC VALUE 1 ON TOB IP) (RC VALUE
112 REQUIRES VC VALUE 1 ON TOB IP) (RC VALUE 111 REQUIRES VC
VALUE 1 ON TOB IP) (RC VALUE 110 REQUIRES VC VALUE 1 ON TOB IP)))
  )
 )
```

In the above excerpt, the payer is designated to be California Blue Cross, as evidenced by the designation CA BC. An edit VALUE CD 1 REQUIRED TO BILL PVT ROOM, which means that if a private room is to be billed to payer California Blue Cross, ________________ **Value Code is a composite code used in the healthcare industry to indicate a service and the cost of that service. In this example Value Code 01 indicates the "Most Common Semiprivate Room Rate, the second element for the value code may be a numeric value which in this case would be the daily rate in US Dollars for a semiprivate room**, is translated to a plurality of translated edits, of which RC VALUE 110 REQUIRES VC VALUE 1 TOB IP is one. RC VALUE 110 REQUIRES VC VALUE 1 TOB IP means __ **Revenue Code (RC) 0110 requires Value Code (VC) 01 on claims whose Type of Bill (TOB) indicates inpatient (IP). .** ________________**.** Although the edit above has been translated to yield a plurality of translated edits, in some cases, a one-to-one correspondence between a given edit and its translation will result.

Once the translated edits have been generated, the translated edits may be used in conjunction with knowledge-base-population rules in order to populate the knowledge base. At step 804 the translated edits are used to populate the knowledge base. At step 806, knowledge in the populated knowledge base is used to validate claims.

FIGURE 9 is a flow diagram that illustrates in more detail step 804 of FIG. 8. Step 804 includes steps 902-910. At step 902, the translated edits obtained in step 802 are applied to knowledge-base-population rules in order to determine when syntax of a given knowledge-base-population rule and syntax of the translated edit match. An edit object that is created for each edit acts as an intelligent agent. The edit object has all of the translated edits that apply to the code context in which the edit object is used. The code object is adapted to apply the translated edits applicable to the edit object to the knowledge-base population rules in order to determine when a syntactical match has been found. Once a syntactical match has been found, the knowledge-base is populated, as described in more detail below.

Each knowledge-base-population rule is given a unique name. Below is an example of a knowledge-base-population rule referred to as code-requires-other-rule-1:

```
(code-requires-other-rule-1
     ((? edit) (? name) value (? value) requires (?
name-2) value (? value-2) on tob (? tob))
     (add-required-code (? edit) (? name) (? value)
(? name-2) (? value-2) (?  tob)))
```

A question mark followed by a symbol inside of parentheses designates a variable (e.g., (? edit)). Therefore, (? edit), (? name), (? value), (? name-2), (? value-2), and (? tob) are all variables used by the knowledge-base-population rule code-requires-other-rule-1. A symbol without any parentheses around it is a constant of a pattern of the knowledge-base-population rule. Therefore, value, requires, and tob all are constants of the knowledge-base-population rule code-requires-other-rule-1. Each knowledge-base population rule represents a pattern that is a combination of both variables and constants in a specified order that is used by the intelligent-agent code objects in a pattern-matching procedure. The above-listed knowledge-base-population rule operates as follows: The rule described above conforms to a discrete structure. The rule has a name, which is, in this case, code-requires-other-rule-1. The rule has a condition statement, such as ((? Edit) (? Name) value (? Value) requires (? Name-2) value (? Value-2) on tob (? TOB)). Many translations may conform to this pattern. For example, <edit object> RC value 0420 requires OC value 11 on tob 14 conforms to the pattern of the condition statement. Finally, the rule has a conclusion, which is, in this case, (add-required-code (? Edit) (? Name) (? Value) (? Name-2) (? Value-2) (? TOB). The conclusion of the rule describes an action that an edit object should take when the condition portion of the rule is determined to be true.

Returning to FIG. 9, at step 904, in response to a match between a translated edit and a knowledge-base population rule being found, the edit object, acting as an inference engine, binds conclusion variables of the matching knowledge-base-population rule to the values of those variables in the translated edit of the code object that resulted in the match. A method of the edit is called to effect population of the knowledge base. In the example discussed above, the method and the conclusion are both designated add-required-code, since the conclusion that a required code needs to be added is effectuated by the method add-required-code.

A particular knowledge-base-population rule will only match assertions for a translated edit that has the exact same pattern as the knowledge-base-population rule. In the exemplary knowledge-base-population rule code-requires-other-rule-1, only if a translated edit matches the pattern `(? edit) (? name) value (? value) requires (? name-2) value (? value-2) on tob (? tob)` will the variables in the conclusion of the rule be bound to the values of those variables found in the translated edit. In the example above, the match between the translated edit and the knowledge-base population rule results. Next, a method (i.e., add-required-code) is called. The method add-required-code is specialized for edit objects whose translations match the pattern of the knowledge-base-population rule code-requires-other-rule-1 and uses name-value pairs and other information instantiated by the add-required-code method. The method add-required-code performs the following functions: First, the method creates instances of the appropriate claim code objects. In the example above, the edit object creates a claim code object named Revenue Code, with a value of 0110. Next the edit object translates the bill type to ensure that all appropriate types of bill are included in the edit (e.g., type of bill IP translates to Inpatient and ultimately to type of bill 11 and 21). The edit object sends a message to the initial claim code object to add a relationship. In the current example, the relationship may be "requires VC 1 on tob 11." The claim code object creates the appropriate relationship with the Value Code object whose value is 1. Finally, the edit object directs the claim code object to insert the newly created relationships into the knowledge base.

When the pattern of a given knowledge-base-population rule is matched by a translated edit, the bound variables listed in the knowledge-base-population rule are instantiated by the inference engine that is processing the rules. When the variables are instantiated, the applicable conclusion of the knowledge-base-population rule is added to a set of edit-rule conclusions. The conclusions of all of the knowledge-base-population rules in which a pattern match has been found are executed via the corresponding methods. Applicable method calls cause SQL queries to be made that cause population of the knowledge base to occur.

Returning to FIG. 9, at step 906, appropriate variable bindings are expanded. As noted above, when a match between a translated edit and a knowledge-base population rule is found, the appropriate variables are bound to the values set forth in the translated edit. At step 906, those variable bindings are expanded so that each of the expanded translated edits (i.e., the plurality of translated edits that correspond to a single edit) includes the variable bindings. Of course, when there is a one-to-one correspondence between an untranslated and a corresponding translated edit, there is no need for any variable-binding expansion as in step 906 to occur.

For example, a single edit translation has the potential to expand into multiple statements of fact that are inserted into the knowledge base. Continuing with the earlier example, the edit "RC 110 requires VC 1 on tob IP" is applicable to inpatient claims. Inpatient claims are represented by Type of Bill codes 11 and 21. In order to ensure that all appropriate types of bill are included in the knowledge base, the abbreviation "IP" is expanded to address all inpatient bill types, i.e. 11 and 21. Thus the single conclusion "add-required-code <edit> rc 110 vc 1 ip" (the result of having bound all the appropriate variables in the rule) is expanded to two calls to the "add-required-code" method as follows:

Add-required-code <edit> rc 110 vc 11; and

Add-required-code <edit> rc 110 vc 21.

At step 907, a translated-edit quality check is performed. The quality check of step 907 includes a check to determine that the variable bindings have been correctly bound and expanded. As described above, embodiments of the invention permit edits to be compared to one another via the contents of their edit strings and determining their level of similarity. In order to verify that the variable bindings have been appropriately bound and expanded the variable bindings, pattern matching may be employed.

The expanded translations are first iterated through using pattern matching with the corresponding translated edit prior to expansion. From the above example, expansion of the edit "RC value 110 requires VC value 1 on tob IP" results in the following two expanded edits:

RC value 110 requires VC value 1 on tob 11; and

RC value 110 requires VC value 1 on tob 21.

If the pattern matcher fails on every instance of the expanded translated edit, it has been verified that at least one modification has been made to each of the translated edits relative to the corresponding unexpanded translated edit. Attempting to match the pattern of the original edit translation "RC value 110 requires VC value 1 on tob IP" with either of the expanded edits fails on each of the expansions because IP does not match 11 or 21. This failure indicates that the original edit has in fact been modified.

The next step is to determine how closely the expanded edits conform to the original edit. The similarity of the expanded translated edits is compared to the variable bindings of the original structure of the unexpanded translated edit so that it can be determined whether a predetermined level of structural similarity exists between the expanded translated edits and the unexpanded translated edit. This is done by creating token lists of the original edit and the expansions. In this example, the token list of the original edit is "RC value 110 requires VC value 1 on tob IP". The token lists created from the expansions are "RC value 110 requires VC value 1 on tob 11" and "RC value 110 requires VC value 1 on tob 21." Taking the intersection of the token list of the original edit and one of the expansions yields a list of tokens, e.g., "RC value 110 requires VC value 1 on tob". This intersection contains nine tokens. There were an average of ten tokens in the original lists used to create the intersection list. Dividing the length of the intersection by the average length of the lists used to create the intersection yields a .9 similarity. This analysis indicates that the process has changed the initial edit and that the result of that change is 90% similar to the original edit. Thus we conclude that the edit expansion has been successful and that we can insert the expansions into the knowledge base.

At step 908, instances of objects (e.g., add-required-code) that include the information bound to the matching conclusions are created. As described earlier, an edit object creates an instance of a claim object and instructs the claim object instance to create the relationship indicated by the elements of the pattern that are included in the rule conclusion. For example, the edit object creates a claim code object with the name "revenue code" and the value "0110." The revenue code is instructed to add the relationship "requires VC 1 on tob 11." The claim code object then creates an instance of the appropriate relationship and requires a relationship with the claim code whose name is "value code" and whose value is "1" on Type of Bill 11. This information is then stored in the knowledge base where it can be used by downstream claim validation systems.

At step 910, the knowledge base is populated with the information contained in the created object instances. The knowledge-base-population rules are used to identify where the knowledge from each of the translated edits needs to be placed in the knowledge base. The called methods add information from the translated edit into the knowledge base in the appropriate place. For example, when the code-requires-other-rule-1 knowledge-base-population rule matches a translated edit, the add-required-code method is called and the bound variables following the add-allowed-code method are added, as appropriate, to an allowed codes table in the knowledge base.

One of the advantages of embodiments of the present invention is that it is not necessary to rewrite the knowledge-base-population rules when edit changes occur. Principles of the present invention permit the knowledge base to be readily updated in the event that changes are made with respect to how a given claim form must be filled out, wherein an update is considered to be a type of knowledge-base population. The translation grammar is generalized so that a translated edit may be created for a given change and the grammar may be used to identify where the translation needs to be placed in the knowledge base.

The previous description is of a preferred embodiment for implementing the invention, and the scope of the invention should not necessarily be limited by this description. The scope of the present invention is instead defined by the following claims.